- I am getting my enthusiasm back bu doing this longer newsletter and have further issues in the mill. Bill

TS Bulletin #1-90

The Newsletter for Low Cost
Home Computing/Orphans
By:
BILL HARMER
97 Ruskin Avenue
Ottawa, Ontario, Canada K1Y 4B3

NOT TO BE CONFUSED WITH THE "NEWS SUPPLEMENTS"... Issue#1-90, May, 1990, 8 pages.

©1990 in the public domain; photocopy freely!

Well here we are again, the first of the our little, reduced page size newsletters to be put out since last year. This size allows us to put more words (about 7,000) in a small and so cheaper package to mail. Since this has been coming out on an intermittent basis for the last three ½ years, a fair amount of material has been accumulated, (some of it republished in photocopy form in books), on disk system theory, Timex-Sinclair ZX-81/TS1000 Tips and Tricks (hardware and software), and on Famous Computers (from the column of this newsletter of that name). A booklet on Z-80 machine language programming (with the ZX-81/TS1000, Sinclair), also has been put together. These should be available from the above address for various sums of money, from about $15, & up, to cover costs of photocopying and mailing, etc. Inquire.

The writer can vouch for the fact that interest in orphan computers survives, since he is still writing programs (in BASIC) for his Sinclair ZX-81/TS1000. It has a disk drive now and 32K of RAM, and is hooked to an amber monitor, but deep down inside, it is still the old, Sinclair 'doorstop' that many loved or hated but often had their first real, extensive exposure to computer programming with. Current projects include training emulators for assembly language (register to register) programming of the 16-bit DEC PDP-11 and 32-bit computers too. Also coming along are experiments in writing the world's easiest to write compiler, (for the ZX-81/Z-80, and written in BASIC).

All in all, the year has seen the plummeting of prices of the IBM compatibles, and even the AT compatibles and 386SX computers have reached $1999 around here. The profitability of the IBM clone market is being squeezed in the same way that we all saw the profitability of the early orphan microcomputers like the VIC-20, ZX-81 and Radio Shack Micro Color Computer (MC-10) first squeezed and then orphaned. By the way, an MC-10 is still sitting here, unused for several years. An excellent, cheap computer, faster even than the Commodore 64 and 6803 based, it lost its niche in Radio Shack's line-up as the Color Computer itself started to drop in price, squeezing the MC-10 out. I bet market researchers are searching now for a new machine!

Will the Macintosh survive? Will the NeXT take off? What will Mr. Trammiel of Atari do to follow up his fathering of the Commodore 64, Atari 520ST, ... next a NeXT clone, or a super cheap IBM clone? Will your local user group survive? What will the '90's hold for us, the amateur computer programmers and hardware hackers? It should be fun at any rate!

Late News: Seattle TSUG reports Jack Dohaney has bought a stock of Zebra disk systems and will be selling them from his new address: 435 Woodward Way, Athens, GA, USA 30606....FDD Express mag. discontinued (A Zebra disk user mag.)....

CONTENTS:

- Files Off The Bulletin Boards (B.B.S.) p.5-6
- Text Compression - Programming Page Feature p.2
- Hardware: Changing The Microprocessor In Your Computer? . . . p.3
- CP/M, The Operating System Alternative to MS DOS p.8
- Hardware Theory: The Humble Transistor p.4
- Repairing Cassette Recorders (For Orphan Computer Owners) . . p.7
- Miscellaneous News and Fixes . . . . p.1 & 8

Update: -SMUG reports a fix for a problem with defective TS2068 video digitizer boards, symptom, video frames go black. Cause: missing jumper to tie pins 1 & 20 of IC 6 (to +5 volts) results in the 74HCT 273 floating. If you bought one of these boards through SMUG contact them on this.

Z-88 -Sharp's has 1 meg RAM's for the Z-88 (& 256K RAM's). M.Stueber, 1-804-730-9697 (usa)

TS2068 -SAMS Photofacts carries a TS2068 Manual in their line of repair manuals which also includes Commodore 64, Plus 4, 1541 disk etc.

LK DOS → -Larken LKDOS disk system tips & utilities often covered in Toronto TSUG n/l -Why not join and get a subscription? Write: 14 Richome Ct., Toronto, M1K 2Y1

QL → -QL Liberator gets speed & multitasking out of QL running SuperBASIC.

(nos as at Mar. '90) -USA BBS #: Time Designs 300b., 8/1/n: (503)824-2658 // Z-88 BBS (703)968-9438, // Sinclair Echo (East) (201)956-7703 (New Jersey) // Chicago TSUG (312)576-7072

CAD → -CADZ, TS2068 pcb/schem. prog: Zunk El., 1419 Ferris, Waxahachie, TX, USA 75165

An amateur publication by Bill Harmer, Ottawa-serving low cost computer users.

m/c Digraph Pack (Text Data Compression Algorithm)

ref: MCPACK IA

The data compression algorithms for 'big' computers don't adapt well for use with the smaller, 8-bit orphan computers. For one thing thing, they are too long. (The ARC program, ZIP, SQ etc. require at least 15-25K). With a 16K or 32K RAM orphan that certainly cuts their utility. They are also all disk based, using space on the disk to substitute for RAM. Finally, they usually use the Lev-Zempel algorithm, which replaces a string that is repeated frequently in the text with a code number. Problem is, on a short text there is not much room in which the writer can repeat the same word or phrase (or part of a word) compared to a 10K or more text file from a disk. For compressing data within say a BASIC database for an orphan computer, even cassette based, another method must be used. The digraph pack routine is one approach and as described here has given compression to 76% of original text size in an actual test (English and unaccented French). That may not seem much compared to the 30% or ARC or 20% of ZIP, but using them, even if possible would require 15-25K of program overhead, while the m/c digraph pack could do it within a program routine size of about 1-2K or less (compression only).

A system of digraph pack was described in Cortesi's 1983 book, "A Programmer's Notebook" (Pub. Prentice-Hall), and complete machine code for CP/M is included. The priciples are simple. Certain two letter combinations (digraphs) are more frequent than others, say the two letter combinations formed from the most frequently occuring letters of the English alphabet (space, 'etoinshrdlu'). In most coding systems, as in ASCII, there is a block of dispensable codes (in ASCII, all numbers, letters of the alphabet, punctuation are in the codes 0-127, and the codes 128-255 are computer specific, usually graphic characters, etc., not required in a simple text file). This block of dispensable character codes in ASCII, 128-255 is the block that has the most significant bit set to one. That leaves you with 7 bits to play with. Why not make the next three bits the code (in octal, 0-7) for the 8 most frequent letters (including space as a letter, since it is the most frequent), and they can thus code the first letter in the letter pair. The next 3-bits after that can in the same way be assigned the codes 0-7 for the 2nd letter of the pair. The remaining, unused bit can be given a special use or be saved to avoid the use an ASCII etc. code that may be taboo for the operating system or compiler (such as an end-of-block code). Rather than use the most frequent 8 letters, the writer has chosen 8 that fit better into a simple m/c algorithm, that can be implemented on virtually any microprocessor, Z-80, 6502 or 6800 included. Of course, if the compression program does not detect a two letter pair that will fit into the 2-letter, one-byte code, it will simply use the same code and letter in the compressed result as in the original, uncompressed text, passing it through from input to output intact.

Writing the program: To avoid writing a lot of machine code, a BASIC loop can be used to take each character from the input text, from the beginning on, just as you would read the text, and feed the letter code to the m/c subroutine from within the loop using the BASIC m/c call (such as USR), and the m/c code routine will put that letter code into a register (the 8-bit, register a of the Z-80/8080 or accumulator for example). The program then needs two other registers (8 or 16-bit, like 6502/6510 IX & IY, index registers, or Z-80/8080 BC and DE). Say the Z-80's DE is loaded with the code (ASCII or whatever the computer uses for it) of 'a' (or in a computer with capitals only 'A'), and the BC with 0. Then to check for the possible letters to be coded in the pair, 'acegist' and space, compare the character in the accumulator (from the original text) with that in the E reg. and if you get a hit you branch out of the routine and go to test for a 2nd letter hit, saving the value in the C register, zero since it will be the octal code for 'a'. After some parameter passing to BASIC, the BASIC program will feed the m/c routine the next letter in the code and take the correct action depending on whether it too is a hit (one of the most frequent letters) or not. Now, in the case that the letter extracted from the text at first is not 'a', the m/c program will test for the next letter, 'c'. It will do this by incrementing the code for 'a' in the E register twice, to make it the code for 'c' and incrementing the C register once, to make it the octal code for 'c' (that is 1). You see now if the octal code is more than 5, and if so branch to the final three character check routine (s,t, and space). Then you compare the character code from the text with the code for 'c' in the E reg. If they are not the same you loop back to the INC DE, INC DE, INC BC. Space, 's' and 't' have to be tested in separate m/c routine steps. LET O$= O$ + CHR$ Y will form the output string, O$, of compressed text. Short and fast m/c indeed!

Changing The Microprocessor In Your Computer (The Z-80 As Example)

Here is something you may not have thought of — can **you** change the microprocessor in **your** computer in order to improve its performance? If you have already thought of it and done some research into it, as the writer has, you will have found out that it is no simple matter to find a winning replacement for any given make of computer.

Why would you want to do this anyway? To make your computer run the programs of say an IBM PC even though it is a Sinclair, ZX-81? Here you get into a lot of problems for the ROM (BIOS) and the memory, etc. architecture as well as the control, expansion bus and input/output lines are all radically different on the two computers. You would have to redesign the computer from scratch and probably would end up with an IBM clone that would have a ZX-81 compatibility card or mode plus a lot of headaches in debugging the whole thing and actually getting it working. Teams of engineers with big budgets usually try this, not amateurs. "What if I just wanted to be able to run a few machine language routines on it in 8088/8086 machine code? Couldn't I just plug in an 8088 chip?" Well the pin out of the 8088 is different from the pin-out of many chips including the Z-80. The 8088 has extra address lines the Z-80 doesn't. What to do with these? Leave the wires unconnected? But then you will read in the 8088 documentation that the 8088 on reset (or start-up) looks for its start-up machine code up in the address map addressed by these extra wires (in high memory rather than memory location 0000 where the Z-80 looks for them and where they are in the ZX-81/TS1000's ROM). And where are the separate address lines A0-A7 on the 8088? They are not separated from the data lines D0-D7 (they use the same pins). More thumbing of the documentation and now you see you need some more chips and circuits to separate them. Once all that rewiring is done (with an over-the-board, extra pc board) then the 8088 will no longer run the machine code in the ZX-81/TS1000 ROM since it is in Z-80 machine language, not 8088 m/c. Those are the sort of steps that make it impractical to use a chip, even of the same heritage (as the 8088 is of the Z-80) to replace the microprocessor in your computer.

What about using a CMOS chip so that at least we could take some of the strain off the power supply? That is a more practical idea in that many CMOS chips are pin-compatible with the MOS versions of the microprocessor. The main problem here is finding one since they are often made by smaller manufacturers so that documentation is hard to find as well as a retail electronics store that stocks the microprocessor in CMOS version. For example the Z-80 has CMOS versions numbered Z80C00 and perhaps other numbers. But again since they are produced by small companies often it is hard to tell if they will work since the writer has been unable to get documentation. V-20 and HD64180 would not seem to be pin-compatible (and the V-20, when switched to compatibility mode takes only 8080, not Z-80 m/c). If you have further information, the writer would like to hear about it.

What about using a faster chip like the Z-80B instead of the Z-80A? Wouldn't that speed up the computer? The answer is no, not unless you increased the clock speed to and in many computers (like the ZX-81/TS1000) that would create a lot of new problems. But using a Z-80B might be a good idea anyway since it may prevent the computer from crashing due to fast glitches, even though it would not speed up your programs at all.

So there are many technical considerations in changing the microprocessor in your computer in order to make it work better and people who have the knowledge and will to solve them usually can make much bigger bucks working for a large corporation doing that sort of work than catering to the orphan computer aftermarket. If you want to try it though as a hobby, first start doing the research through the books that are relevant and then even if you decide not to go ahead you will have learned to better appreciate how your computer really does work inside!

The Hacker's Info Sheet: The Humble Transistor

One of the most neglected tools in the hardware designer or hobbyist's box of tricks is the single transistor amplifier used as an on/off current switch. It acts like a relay, taking a small current and using it to switch a much larger one.* How much larger? Well, since the whole principle of the NPN or PNP transistor's amplifier is specified in terms of its minimum beta (β) or hfe (which is the same thing, but in fancier jargon, that of the 'h-parameter', electrically equivalent schematic model of a transistor) and the beta, alias h_{fe}, is simply the current amplification, the current a transistor can put out depends on how much you give the base to drive the transistor with. Current out= beta x current in (at the base). Well, that isn't quite true when you drive the transistor to the max (saturation, in the jargon), for then some inefficiency occurs and you will need a little more input current to get it to the max.

So, you can use a little transistor like a 2N2904 or 2N2222A or 2N4401 (to mention the three most popular little NPN transistors for hobbyists and designers) to step up the current output of an integrated circuit since the MOSFET or CMOS ones, like the 4000 series CMOS logic or CMOS 555 even, are not capable of much output without blowing themselves out. It can drive an LED, and eprom burner or piezo alarm. Just remember not to tax it with too much power either, or the transistor will burn up itself (power=current {collector through emitter} x voltage of your power source, as a rule of thumb). Try to keep it down to 5 or 10 ma. for the above-mentioned (and nearly all small) transistors. They will take up to 200-300 milliwatts maximum but they won't last to long with that sort of abuse and this little transistor size likes current outputs under 2 ma., the smaller, the better, if you don't need the larger currents. (*Mechanical buzzers use about 1 watt+& a big transistor).

So, how do you keep them from blowing up from too much output current? One method is to work backwards, and limit the input current, since you that the output current is the input current to the base times the beta (hfe). Problem is that doesn't really work in practice, although limiting the input current with a resistor is important anyway so you won't accidentally blow the transistor from the front end (the base-emitter junction, which is like a diode through which current flows, the input current plus the rest of the current).

So as our adaptation for this design proceeds, we must find a way to limit the output current. The usual way to limit the output current of anything, a transformer, battery, or anything else is through the load attached to it. If that isn't enough of a resistance to limit the current far enough, a resistor can be placed in series with the load, to reduce it further. There are other fancier ways of reducing the current output of a transistor, since the transistor is a fancy device and an unusual one in itself (as well as its applications). But let's stick with the resistor method. Current out= (at maximum) the voltage of the power supply divided by (/) the sum of the resistances blocking it. (Ohms Law simply applied to the problem , I=V/R).

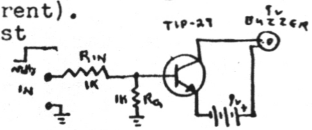

Fig. Buzzer Driver*②

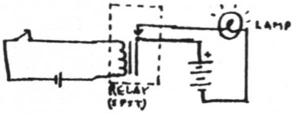

Fig. Relay†

So a transistor, used like this, to step up the current output from an integrated circuit, acts like the older computer technology of relays.* A relay uses a small current to magnetize an electromagnet which pulls a switch bar over, to switch on a larger current (since the switch it turns on -or off- when the electromagnet is magnetized, {and it is magnetized as long as current is passed through it, via the input}is designed to handle a larger current -or voltage- or both). So it acts like a relay,* and indeed it may be used to step up the current of an i.c. so that the i.c. may work a real relay and shut it on and off. Just remember to include a diode in the design of your relay driving transistor amplifiers. When the transistor shuts off the current to the relay input coil (the coil of the electromagnet that tugs at the switch), a voltage snap-back effect occurs that will zap the transistor, burning it. (Voltage snap-back=-di/dt, the derivative of a square wave! for calculus buffs)

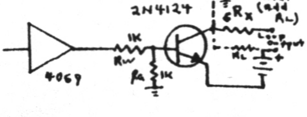

Fig. CMOS Output Amplification①

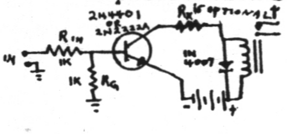

Fig. Relay Driver①

NOTES:
① = circuits, untested, Not guaranteed
† = requires right specs.

*Of course the relay adds a safety factor, isolation, the transistor doesn't. //End

Files Off The Bulletin Boards

Now that more and more people are connecting their orphan computers to modems and capturing text files and other things (even source code programs in BASIC or Pascal), many are starting to notice peculiarities not to say partial incompatibilities in what they are getting in their computer relative to what the computer's own keyboard will put there.

The lingua franca or universal language of computer communication is ASCII. Too, most but not all computers, even orphan ones use ASCII as the base for their alphanumeric code. The computer's internal code assigns an 8-bit (byte size) number to all the relevant characters that will print on the screen plus a number that won't. A byte of eight bits represents the numbers 0 to 255. Typically, 'A' will be 65, 'B' will be 66, 'C' will be 67.... To get an 'A' inside your computer from the keyboard, you will press the 'A' key and maybe also the capitals shift key. After that computers diverge. Some will allow combinations of keys like control-'C' and so on, to get the other characters in and some computers will display these key combinations using peculiar symbols on the screen. Some computer makes have unique keys (like the Commodore key on the Commodore 64 or the Gold key on DEC VT-100 terminals). The Commodore has two sets of characters and when in the mode for set 2, the 8-bit number 65 is 'A', while in set 1, mode it is the card-player's spades symbol(♠).

Somewhere in the software and hardware design of your computer, the designer had to come to grips with representing ASCII characters with your computer (on the screen as well as via the printer). Some computers do not use ASCII at all (IBM 360 mainframe uses EBCDIC, Sinclair ZX-81 uses its own code). The ones that do incorporate some compatibility with ASCII into their internal codes usually do so on the basis that ASCII is only a 7-bit, 0-127 in 8-bit numbers, code, so that the computer's own code having an extra (most significant) bit can have the codes 128 to 255 (128 more possible characters) to use in its own unique way. The IBM PC for example uses these codes for a lot of graphic characters like the frame parts that can go around a program title when you first start up the program. The Commodore 64 uses as mentioned a more complicated doubling up of the codes so two character sets, regular ASCII more or less and Commodore, game relevant signs and symbols can share the first 128 codes (0-127), and leave all the codes above 127 (128 to 256, that is those with the most significant bit turned on) for the reverse video versions. However, there are various possible sets of coding systems. The Commodore has separate display code and data/character code. Many computers tell you a third type of code, the scanning code (the number fed to the computer by pressing a certain keyboard key) while some only bury that information in technical material for the machine language programmer. Each of these three possible sets of codes, video, keyboard-character, and scanning codes are used by a different part of the computer's built-in operating system. The easiest way to see the set that you can manipulate with the BASIC programming language is to write a simple BASIC program: 10 FOR X=1 TO 255 , 20 PRINT CHR$(X);"-";X, 30 NEXT X. This will give you something to think about to start. The other code systems may require POKE's or PEEK's to get to from BASIC. The display code for example may need be POKE'd into the display RAM.

To get standard ASCII converted to your computer's code in text files, a simple BASIC program may be all that is needed. You may wish simply to throw away any characters with codes, CHR$(X), above 127, X>127. To convert ASCII to the coding system of an orphan computer without any lower case letters, just convert the CHR$(97 to 122) range of lower case to the range of your capital letters (usually, but not always, CHR$ 65 to 90). Try this: Take the ASCII code of each character by say ASC(C$),(where you take each letter of the text in order and make it equal to C$, by LET C$=MID$(L$,Y,1) -if you have MID$-, L$ being the whole line and Y being at the letter in the line, the Yth letter from start of the line). Next step, add the right number to it or subtract it, in this case 'a' with code 97 can be made into the code for 'A' by subtracting 32 to give the code 65 of 'A'. Then print the character on the screen with CHR$(X), where X= now 65, that is 97-32. Other characters may even have to be converted

..... → p.6

5.1-90/A-90

by individual statements like IF ASC(C$)=36 THEN PRINT "$"; while other characters in ASCII that can't be displayed might be printed as '?' or '#' or something to give you a clue that a character is there but not a normal one, for your computer's set of displayable characters.

In text files there are however certain ASCII character codes that have special meanings. For example CHR$(26), called 'SUB' in the ASCII reference charts, is used in early to middle versions of MS DOS for the IBM PC,AT etc. to indicate the end-of-a-text-file (some books refer to it as control-Z since that key combination on an IBM PC will enter that code at the end of the text file). This was probably carried over from CP/M which uses the same character for the end-of-file code. Later versions of MS DOS do not require it but are not bothered by reading files ending in it. So your computer should, if it is not already in its operating system, be programmed in your file translation program to detect the end of a text file read in from a BBS by looking for CHR$(26).

The other thing to look out for is the end-of-the-line indications. Most computers end with a carriage return, no matter what it is called on the keyboard (or even what code is used). Some computers call it ENTER, come RETURN, some Newline, some ↵ or a similar symbol, and some CR or Carriage Return. In ASCII that is CHR$(13) but in some computers another code may be used (Sinclair ZX-81 uses CHR$(118) which would translate to small letter 'v' in ASCII !). The standard IBM PC etc. uses it in MS DOS operating system generated text files as the end-of-line indicator so that each line (up to 128 characters long) ends with a CHR$(13) to indicate that what is beyond that is a new line, the next line of text. However that is not universal among operating systems, and CP/M, once designed to use telex terminals as input and output uses CR, LF (the latter is CHR$(10) or line feed) since telex terminals needed to be told to return the the print head or paper carriage to the starting position at the left hand side of the paper and also to to move the paper up one line (so as not to print over the last line of typing) and this is the line feed action. So with CP/M generated text files, which will of course be ASCII, CR will be the second last character in the line and line feed (LF, CHR$(10)) will be the last character on the line. Note that although different computers have different line lengths of screen display (32 character, 40 character, 64 character, 80 character) both the CP/M and MS DOS operating systems (and a lot of programs like modem programs and wordprocessing programs), use a maximum line length of 128 characters.

What we have been talking about is text files, like documentation (xx.DOC), telling you how to use a program or text files (xxx.TXT) telling of news or views and ASCII files as they are called in some computers (xxx_ASC, for example), that may do either. Programs in BASIC or Pascal may, in source code form, often be saved in ASCII and then transmitted that way but compiled programs (xxx.COM) and other machine code programs will not be translatable as described in this article. They are not text or ASCII files.

Compressed text files like the xxx.ARC and xxx.ZIP files and the older ones which take the middle letter out of the extension and substitute a 'Q' making xxx.DOC into xxx.DQC and xxx.TXT into xxx.TQT, are another matter and must be processed by an de-archiving or expanding program before they can be converted from ASCII to the form of another, say orphan computer. Programs to do this are available for MS DOS and CP/M but not many other computers have them, although the source code for translation from compressed form is often available in C language and the programs when compiled (C language must be compiled), usually take up about 20K to 30K of memory, making them beyond the capabilities of a 16K orphan to handle. 64K computers might have a chance of using them if a hobbyis can be found to translate the C language source code to a dialect suitable for a C compiler that will run on the orphan computer, and compile and test the program. The writer knows of no BASIC programs to expand compressed files and of only one or two Pascal programs (one for the SQ,UNSQ system mentioned above which makes xxx.TXT into xxx.TQT files, and is considered obsolete — and a pair for ARC, available through Turbo User Group). ///LNL

g.90.1/8-1-90/A-90

Repairing Cassette Recorders

When cassette recorders used with home computers stop working, the hardware hacker may think that there is nothing that can be done but buy a new one. Actually in some cases, fixing them is not too much for the handy, electronics hacker.

This of course assumes that you have elimintaed all the other potential problems, and had the system working before it quit working. New users of cassette systems like the temperamental Radio Shack Model I, the cantankerous Sinclair ZX-81 and Timex 1000 and the RFI sensitive Commodore Datasette (which some say has to be placed the full length of its interface cord from the C-64), who are having problems maybe are best referred to the magazine and memory databanks of of veteran users at user groups. It may be merely a matter of finding the exact volume (and or tone) control settings. Other variables are the cassette itself (is it a fresh one, or has it been used before, a factor that in the ZX-81/TS1000 may make it unusable), and whether it has, once the recording has been made on it been placed on something hot (like the monitor/tv) or something magnetic (like the top of the cassette recorder itself, whose speaker has a powerful magnet in it, close enough to the external case to cause trouble). Alternately, the computer's circuits themselves may be damaged. Unplugging or plugging in the interface cable while the computer is trying to save or load a program, may ruin the part of the(MOSFET) static and spike sensitive integrated circuit(s) that are a part of most home computers and to which the cassette often directly connects! Assuming that none of these are problems and you have cleaned the heads with head cleaning fluid and a Q-Tip or some similar swab, and have demagnetized the head (Radio Shack #44-1165 is an example of one type of demagnetizer), then read on. (The other cassette problems have been covered well enough elsewhere.)

There are several symptoms of problems. One is lack of volume, particularly intermittently. The commonest cause for this and intermittent playback or recording (either) is the multi-contact, record/play switch. It is usually somewhere on the printed circuit board where a mechanical linkage can move a plastic (and square) contact rod inside a metal (tin-sheet metal) sleeve. This often gets dirty and represents the commonest problem with cassette recroders, of an electrical nature at least. It may be hard to restore proper contact however, the cleaning with solvents such as tuner cleaner, often having to be repeated and the manipulation of the rod, in and out, while spraying, tedious. Don't be surprised if the first attempt doesn't get it clean and restore the contact action to what it should be for trouble-free operation.

A second problem is that the input to the recorder from the mic. jack is damaged electronically. If it is an integrated circuit, forget it. Getting a replacement is probably not economical and will lead you on a wild goose chase through suppliers. But most have transistors here usually one or two small npn (TO-92) type ones that can easily be replaced by anyone who can replace such transistors in a t.v. repair or radio repair. Typically a 2N2222A or MPS-2222A or PN2222A or 2N4401 will do as a substitute. The faulty transistor may have leakage rather than straight failure and therefore may not be testable with a simple transistor tester (that is it will test good when it ain't). This transistor problem is the second choice, in the 'pitch until you hit it' method of troubleshooting a cassette recorder with a volume problem.

Power supply problems (like low batteries of all things!) can also cause problems, and should be the first thing to check if you are running off batteries or an adapter (that perhaps has been left on too long?). Of course if the mechanism is jammed and will not wind or cause the tape to move during play or record you may well have a mechanical problem with the recorder (or maybe even just with a baulky cassette). Mechanical skills here with cleaning and lubrication can solve the problem. You may also mistake for a mechanical problem, which is eating tape (causing it to disappear into the inside and wrap around things), the fact that the little black rubber roller that pulls the tape through, is encrusted with enough brown tape coating to cause the tape to stick to it momentarily instead of coming off cleanly, and once getting loose, wrap around parts of the mechanism, the tape being that way, 'eaten' by the machine.

It is hoped that this list of a few of the common problems and fixes that the electronic technician can use, may help narrow the search for cassette recorder problems of the home computer user.

CP/M On Timex-Sinclair Computers

There are older generation computers that used CP/M such as the early KayPro and the Osborne portables, as well as the early convertibles like the Radio Shack TRS-80 Models I,III & 4., Apple II and Coleco Adam. KayPro and DEC(Rainbow) made dual CP/M and MS DOS computers at one time. Amstrad made CP/M computers before their MS DOS ones. But few amateurs are aware that virtually any IBM compatible can run CP/M via a public domain software emulator (Z80MU) and many Timex-Sinclair owners are not aware that lurking inside their QL, or TS2068 with Aerco disk system, is a CP/M computer that only requires the addition of some software. The Aerco TS2068 disk uses RP/M, a CP/M compatible operating system, as an option, and a CP/M emulator will allow the QL to run CP/M programs.

What does CP/M have to offer? Some compilers that have been placed in the public domain for one thing. DBase II (the predecessor of the more famous dBase III database standard), -not in the public domain, however. Turbo Pascal, Wordstar and other name brand wordprocessors, and many once famous spreadsheets, and application programs, of all sorts.

The CP/M world, still suported by shareware distributers and clubs like FOG, First Osborne Groups, offers yet another realm for the curious amateur to explore and the writer of this column too is just starting to unearth the pay dirt in this once, expensive to enter world of microcomputer tinkering. Don't be put off by the old, myth that CP/M is hard to learn. Anyone who has a bit of MS DOS experience can tell you that it is easier than MS DOS, and remarkably similar and so quick for the MS DOS user to grasp.

BBS/Modem Hacking

Bought a modem and can't get it working? Is it your first modem? If so, it may not be the fault of the computer, the modem or the software. Noisy telephone lines in some locales can render modem communication, BBSing, especially in the time of day when offices are open, impossible. Bit error messages on the screen are a common symptom on some computer types. There are plans for a do-it-yourself fix circulating in the computer community involving a couple of high voltage, 0.02 ufd capacitors, but it is not known if the phone company would object to this attachment to your phone line cord, from computer to phone outlet.(refr.: NOISE-3.ZIP file)

CP/M Software Sources

FOG, P.O. Box 3474,Daly City, CA, USA 94015-0474 (First Osborne User Gp.)
NLG,P.O. Box 81, Shelbyville, IN, USA 46176 (Tel. 317-398-2370 -all disk formats
PD User Group,Box 1442, Orangepark, FL, USA 32067 - all disk formats
Ottawa Home Computing Club, P.O. Box 4164, Station'C', Ottawa, Ont. Can. K1Y 4P3
(Some of the club's CP/M software is on their BBS)
Z80MU and other CP/M emulators for MS DOS: Computer Parts Galore, 316 College St., Toronto, Ont. Canada (928-2161)
CP/M emulator for Sinclair QL: American Micro Connection, 2175 Aborn Rd., #262, San Jose, CA, USA 95121 (408) 270-9730

Note*Zebra 64K disk interface from J. Dohaney (see p. 1) for Sinclair TS2068 is CP/M compatible.

Read Me First:

Answers To Questions-

Where does this newsletter come from? It is the result of one person's modest efforts to support low cost computing by offering documentation and news and views of interest to those who have found that you don't need a lot of money to have a computing hobby, but have found a lack of supporting publications for the old, inexpensive orphan computers....Can I subscribe? No, but inquire anyway, especially if you wish to buy back issues (for about $2 each, minimum $6 per order). Why don't you put this newsletter on disk? Unfortunately there is no common disk format that all orphan computers can access (and many do not use disks, just cassette). Is this newsletter free? Nothing in this life is free. I make some copies, paying out of my own slender means, in the hope that others, especially user group members will make more copies.------- It is prepared as a hobby by amateur publisher and computer wizard/bumbler, Bill Harmer, 97 Ruskin St.,Ottawa,Ont.,Canada K1Y 4B3 and is sent to a number of user groups in hopes of getting a sample newsletter exchanged. Of course, there can be no guaranty as to the accuracy or completeness of any info or addresses, etc. in this. Warning: make sure you know what you are doing and have a second source of qualified advice to double check by, before you hack expensive stuff.